A System for the Design and Development of Vision-based Multi-robot Quadrotor Swarms

Jose Luis Sanchez-Lopez, Jesús Pestana, Paloma de la Puente, Ramon Suarez-Fernandez and Pascual Campoy

I. Introduction

This paper presents a framework designed to accelerate the prototyping of successful multi-aerial-robot behaviors for the research and development of civilian applications of small Unmanned Aerial Vehicles (sUAVs). The motivation of this framework is to allow the developers to focus on their own research by decoupling the development of dependent modules, leading to a more cost-effective progress in the project. In order to achieve this, the framework has been made public and open-source, and the basic instance of the framework offers several open-ended modules required for experimental multi-aerial-robot navigation.

The basic instance of the framework was designed to ease some of the main issues related to working with aerial multirobot systems, such as: localization, obstacle avoidance and partner detection. Our focus is to allow the team to work in parallel so that it is possible to advance in late stages of the project without depending on the development of the initial stages, such as: hardware platform, state estimation, flight controllers, etc. It is possible to test new multi-robot behaviors first in simulation and, afterwards,on experimental flights based on the cost-effective testbed Parrot AR Drone 2.0. Experimental flights using another desired quadrotor platform can be executed once an interface compatible with the framework has been implemented. This way the advantages of novel strategies can be experimentally demonstrated on early stages of the project development.

In order to achieve the discussed interchangeability among hardware platforms, and in order to avoid the costly acquisition of a motion capture systems such as Vicon4, the following design decisions were made: first, the localization problem is simplified by means of visual markers; second, each robotic agent broadcasts its current estimated pose; and third, the basic instance of the architecture can fly the AR Drones 2.0 with full obstacle avoidance capability. Note that the framework was designed to work with quadrotor UAVs.

Fig. 1. Experimental flight of the basic instance of the framework, which is characterized by swarm behavior, using 5 AR Drones. The row of columns represents a virtual wall with a single 1 5 m opening in the middle. In order to simplify the localization problems the columns are marked using ArUco visual markers [2]. In this flight the mission of the swarm is to cross from one side to the other, each swarm agent also has to avoid collision with the columns or the other drones. This test was designed to showcase the capabilities of the basic instance of the framework.

The second motivation to design this framework is to test the use of aerial visual multi-robot system for civilian applications. There exist a large variety of applications which require a robotic system to densely navigate a wide area. Such applications can benefit from a swarming approach for the required data gathering. For instance, such an approach could be applied to search and rescue solutions.

This paper is intended to present and describe the designed open-source framework. The layout of the paper is the following. First, a description of the characteristics of the framework is discussed in section II. Second, a thorough

description of the basic instance of the framework is done in section III. Third, In order to showcase the practical capabilities of the architecture, several simulation results are presented in section IV, followed by corresponding experimental flights presented in section V. And lastly, the future work and the conclusions are discussed respectively in the sections VI & VII.

II. System Description

This paper presents a fully-functional framework designed to ease the development of multi-sUAV autonomous systems, with an special emphasis on vision-based quadrotor swarms. The main design specifications of the system are the following:

- Modularity, to allow code reuse for different solutions. For this reason, the Robot Operating System (ROS) is used as middleware between modules and across computers. To better understand the characteristics of ROS, see [15], [12].
- Compatibility with various quadrotor platforms through the usage of a well specified interface. The use of visual markers and the choice of AR Drone 2.0 as first compatible platform were done to obtain cost-effective solution.
- Capability of realizing multi-aerial-robot missions. The robots would be connected through WLAN, see figure 2, and they would communicate under the ROS middleware.
- Flight-proven and capability of running simulations on big parts of the developed architectures.
- Open-source, so that we can share our works and other developers can reuse any part of the architecture in their own projects.

Our framework uses the Robot Operative System (ROS) [5], a worldwide used API that eases the management of communication between the software modules of our system. The framework modules were programmed in C++ using the most recent version of the standard of the C++ programming language, C++11. In general, the ROS communication interface has been separated from the main functionality of the modules by means of wrappers.

The key characteristic of our framework is modularity. This allows to create independent modules with specific functionalities but that can be exploited once connected to the rest of the architecture. This modularity allows the individual testing of modules easing the project progress. Also, understanding the modules as input-output systems permits to test in simulation the compatibility of their interfaces with the full system instance at hand.

The compatibility with various quadrotor platforms is achieved thanks to the modularity requirement. Since each module is defined by its interface, different platforms can be used with the only requirement of respecting the specified interface. However, if a platform is heterogeneous with respect to every supported platform (i.e. it uses different sensors), the developer can leverage from the modularity requirement to minimize the required work of interfacing

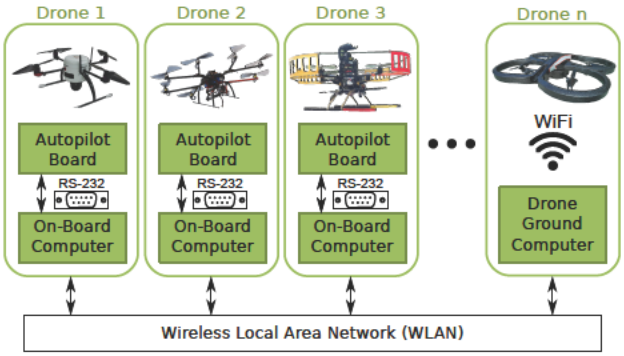

Fig. 2. The multi-robot system is composed by robotic agents, which consist of a quadrotor platform and an specific instance of the software architecture. The drone is either commanded via WiFi from a ground station or from an on-board computer. All the computers can communicate with each other through a Wireless Local Area Network (LAN). The communications between modules and robotic agents is implemented creating a single ROS network.

it with the rest of the architecture. To achieve this goal, the interface between modules has been specified and each robot agent uses its own configuration files. Each module can be executed in an on-board computer or in ground computers. However, all the computers have to be connected in a local area network (LAN) or in a wireless local area network (WLAN), see figure 2.

The framework is fully operative, which is shown in the paper through simulations and real flight tests of up to 5 drones, and was demonstrated with the participation in an international micro-aerial vehicles competition. Since we trust in our system and we believe in sharing with the scientific community, we decided to make our framework open-source. This way, anyone interested in working with multi-robot aerial applications can use our framework as a starting point of their research and as a tool to test their own algorithms. The link to the framework's code Git repository is specified in the following website: `http://www.vision4uav.com/?q=quadrotor_stack`.

III. Modules

As said in section II, the proposed system framework has the main characteristic of being modular and every person interested in working with this system is able to create new modules and have it functioning easily. In this section, our particular module's implementation is defined. Previous works of the authors in this framework are described in [25].

In figure 3, our modules of each agent of the swarm, and the way that they are connected are represented. The modules are divided in five big groups, described in the following subsections:

- Drone and Low-Level Control, section III-A.
- Localization and Mapping, section III-B.
- Mid-level Control, section III-C.
- High-level Control and Intelligence, section III-D.
- Human-Robots Interface, section III-E.

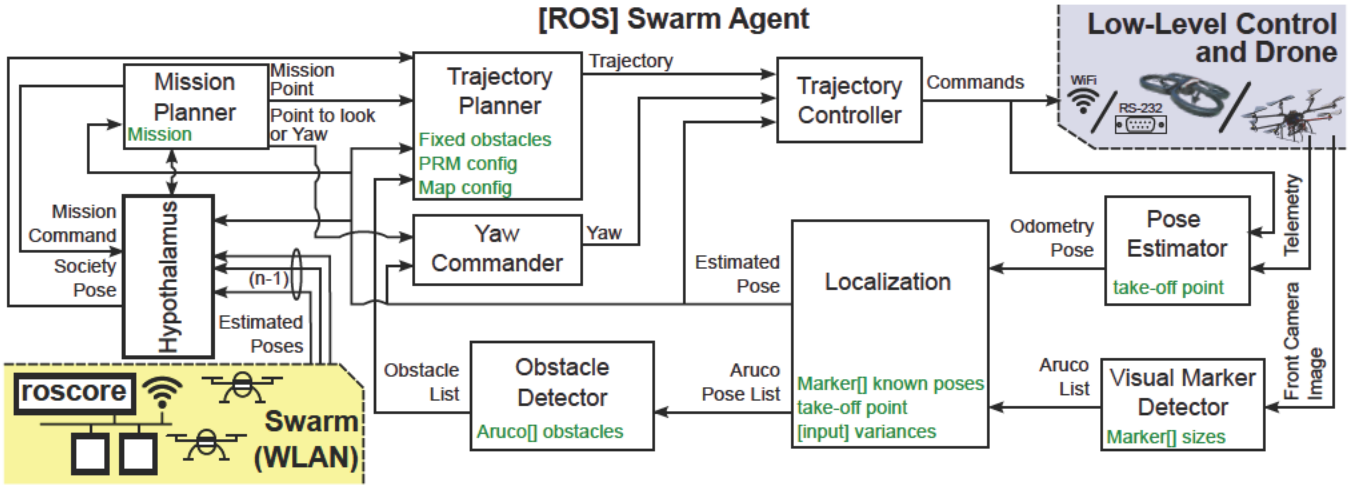

Fig. 3. Our Implementation of the Modules of the Framework. The architecture is modular and is built using the Robot Operating System (ROS) framework. Each white box represents a module, and the green text inside it are configuration parameters. The localization module fuses the odometry based estimation with the visual markers feedback. This module broadcasts the estimated pose to the mission and trajectory planning modules, to the controller module, to the obstacle detector, and to the other robotic agents. The trajectory planner gives free-collision trajectories to the trajectory controller which also receives yaw commands given by the yaw commander. The trajectory controller generates commands to the drone. The mission planner module monitors the mission given mission points to the trajectory planner. The hypothalamus module receives the estimated position of the other robots and communicates it to the trajectory planner.

A. Drone and Low-Level Control

This module represents the real drone used in the presented system. The drone is formed, not only by the mechanical and electrical components, but also by a set of sensors and low-level controllers:

- Low-level controller that allows the trajectory controller (section III-C) to send commands of yaw, pitch and roll.
- Altitude controller that allows the trajectory controller to send height commands instead thrust commands.
- Take-off, landing and hover controllers that allows the hypothalamus (section III-D.4) to send this kind of high-level commands.
- IMU measurements (accelerometers, magnetometers, gyroscopes) and ground speed measurement, that allow the Odometry Pose Estimator (section III-B.1) to calculate the pose of the drone using the odometry measurements.
- A front camera that allows the Visual Markers Detector (section III-B.2) to see the visual marks that the environment has.

There are no limitations in the multirotor platforms used as long as the requirements cited above are fulfilled. In our particular implementation, we used the cost-effective Parrot AR-Drone 2.0 [4] as the robotic platform. The characteristics of this platform are thoroughly explained in [6] and satisfy all the requirements. The ground computer communicates with the drone via Wi-Fi using the ardrone_autonomy ROS package [1].

B. Localization and Mapping

Localization in indoor environments is a challenging task for UAVs, especially if a low cost and very lightweight solution is required [20], [24], [17], [14]. In the absence of GPS and laser sensors, visual approaches are very popular [24], [17], [14].

In the proposed system, the global localization of each drone is based on IMU and optical flow data for the pose estimation, calculated by the Pose Estimator module. However, this measure has some drift which may be significant, so it should be corrected with more reliable information from the environment when available. Visual markers (see section III-B.2) are used for this purpose to recover the 3D pose of the front camera of the drone with respect to each visual marker.

Since the environment can be partially known in advance, some fixed landmarks are employed. This previously known landmark can be also attached to the previously-known pose and shape obstacles. Other visual markers have to be added to the previously-unknown obstacles. The only requirement is to know which visual markers are attached to each obstacle.

Localization with visual external aids for UAVs has been recently proposed in other works [24], [17], [14]. The method presented by Jayatilleke and Zhang [17] requires all the landmark poses to be known a priori and only works in limited areas, making use of quite a simple approach without filtering of any kind. The work by Faig *et al.* presents an interesting approach for local relative localization in swarms of micro UAVs, that requires to keep external markers always visible. Our method was mainly inspired by the work by Rudol [24], but our models and formulation are quite different from those proposed by Conte [7].

1) Odometry Pose Estimation: The "Pose Estimator" is explained in the following article and Master's Thesis [23], [22].

2) Visual Markers Detector: To correct the drift that the Odometry Pose Estimator module has, absolute measures provided by visual markers are used. Our system employs

ArUco visual markers, which are shown in figure 1. These visual markers are open-source and the software library can be downloaded from [2]. This library allows to compute the current 3D pose of the camera with respect to the ArUco markers that are visible in the current frame. The visual markers help us solve two problems at the same time in quite a straight forward manner: the problem of sensing the various obstacles, which is a very hard task using only computer vision techniques, and we also avoid the visual localization problem in a general environment, which has not been entirely solved yet. A remarkable work where visual localization is achieved using AR Drones is [13].

3) Localization and Visual Markers Mapping: The inputs of this localization module are hence the pose estimation result (similar to odometry) and the relative observations of the visual markers.

We designed and implemented an Extended Kalman Filter (EKF) that allows the complete 6 DOF pose of the drone to be corrected by integrating the odometry data and the information from the visual external aids detection. The localization method benefits from the existence of known landmarks, but it also incorporates unknown detected features, using a Maximum Incremental Probability approach for building a map of 6 DOF poses corresponding to visual markers positioned in the environment. Similar methods for ground mobile robots were developed in previous work by de la Puente *et al.*, initially based on the observation of 2D point features with a laser scanner [9] and later based on the extraction of planar features from 3D point clouds generated by a tilting laser scanner [11], [10].

In this work, the data association problem does not have to be addressed, since the visual markers readings provide unique ids for the observations and the landmarks. This way, loop closure is facilitated and enhanced robustness can be achieved with a not very cumbersome algorithm which showed nice empirical results in our initial tests.

Non linear state and observation models are used. At each iteration k, the prediction of the pose state x (6 DOF) is given by:

$$\tilde{\mathbf{x}}_\mathbf{k} = f(\mathbf{x}, \mathbf{u})_{\hat{\mathbf{x}}_{\mathbf{k-1}}, \mathbf{u}_\mathbf{k}} = \hat{\mathbf{x}}_{\mathbf{k-1}} \oplus \mathbf{u}, \tag{1}$$

where the $\oplus$ operator corresponds to the composition of relative transformations in the 6D space. The noise in the odometry measurements is considered as Gaussian white noise (as required to apply the EKF), and the odometry increment $\mathbf{u}$ is represented as $\mathbf{u} \sim N(\hat{\mathbf{u}}, Q)$.

The observation model is defined by the following innovation vector for an association of observation $\mathbf{o_i}$ and map landmark $\mathbf{l_j}$:

$$\mathbf{h_{i,i+5}} = \tilde{\mathbf{x}} \oplus \mathbf{o_i} - \mathbf{l_j} \tag{2}$$

The correction of the pose state is obtained by the update equation:

$$\hat{\mathbf{x}}_\mathbf{k} = \tilde{\mathbf{x}}_\mathbf{k} - W\mathbf{h}_\mathbf{k} \tag{3}$$

where W is the Kalman gain matrix of the system. The covariance matrices are updated at each stage of the filter as required [26].

The environment is assumed to be static except for the presence of other drones. The accumulation of drift error if the drone is not able to detect visual markers all the time may require the incorporation of a forgetting mechanism so that the drone can navigate safely with local maps. In our tests thus far this has not been necessary due to the addition of extra visual markers over the floor, but this should be further investigated.

The input parameters of the algorithm (initial pose, covariance values, global poses and ids of the known landmarks) are read from an XML file, by means of the pugixml library [18]. The corrected absolute pose of the drone and the list of global poses of the landmarks belonging to the map are obtained as output of this module.

4) Obstacle Processor using Visual Markers: Once the position of the unknown visual landmarks is obtained, they are processed in order to obtain higher level geometrical features in 2D to be used as obstacles by the trajectory planner. The map of obstacles is rebuilt at every iteration.

To do so, some prior information is required. Each of the obstacles is given a unique id and the ids of the visual markers belonging to it are provided. The poles are modeled with circles given by the coordinates of their center and the radius $c(x_c, y_c, r)$, while the walls are modeled with rectangles given by the coordinates of the center, the width and the length $R(x_c, y_c, w, l)$.

Given the observation of a landmark $\mathbf{l_j}$ belonging to pole i, an initial estimate of the circle i is very easily obtained:

$$(x_{c_i}, y_{c_i}) = \mathbf{l_j} + r\mathbf{dir} \tag{4}$$

with $\mathbf{dir} = (cos(yaw), sin(yaw))$. This initial estimate is further refined by the mean value of incorporating subsequent landmarks belonging to the same pole.

A similar algorithm is used to obtain the rectangle models of the wall.

C. Mid-level Control: Trajectory Controller

The responsibility of this module is the transformation of the trajectory references given by the trajectory planner, and the yaw references given by yaw commander in low-level commands of the drone, that allows the drone to follow these references appropriately.

The "Trajectory Controller" module is explained in the following article and Master's Thesis [23], [22]. The controller is able to follow yaw commands while executing a trajectory.

D. High-level Control and Intelligence

In this section, the high-level control and the basic intelligence that each drone has is described. This high-level control is formed by a trajectory planner that creates collision free trajectories, a yaw commander that generates yaw reference to look to the points of the map with more information, and a sequential mission planner that allows the drone to follow a user-predefined mission. The basic intelligence that the drone has is concentrated in the hypothalamus that monitors the state of the drone and its modules, and manage the basic functionalities of the drone.

1) Trajectory Planner and Collision Avoidance: The objective of this module is the computation of a free collision 2D trajectory (horizontal coordinates x and y) to achieve a mission.

This module works as follows: a free of obstacles Probabilistic Road Map (PRM) [8] of the 2D map is generated offline. The advantage of using a PRM instead of a fixed-cell decomposition is that you can select the number of nodes in the graph and their neighborhood. Also, if the robot is moving through a zone with a lot of obstacles, new nodes can be added.

Once the free of obstacle graph is created, an A-Star algorithm [21] searches the path using a potential field map function as cost of the algorithm. This potential field map is built as a sum of one component that attracts the drone to the end of the obstacle zone and another component that repels the drone from any obstacle. The usage of a search algorithm (A-Star) instead of the potential field map alone [19], avoids the problem of the local minimum blockage.

Three kinds of obstacles are considered. The first type of obstacles are the fixed and previously known obstacles which are set during the module startup and are obstacles and never change their previously known position. The second type are the fixed and unknown obstacles that are received from the obstacle generator whose position could change over time, depending on how precisely are the visual markers pose is determined by the localization module. The last type are the unknown and moving obstacles that are other drones and are only considered in the path planning if they are close to the drone. Other drones' positions are received through the hypothalamus module.

Once the path is calculated using the A-Star algorithm, it is post-processed in order to obtain a shorter and more direct path, avoiding the noise produced by moving the robot from node to node of the PRM. The post-processing is done using the value of the potential field map.

When new drones' poses or new obstacles' positions are received, the planner checks if the new obstacles are outside the planned trajectory and if the drone is following the path. Otherwise, the trajectory is re-planned.

With this algorithm we solve the problem of the path planning and the collision avoidance, being able to navigate safely in the map using the Trajectory Controller module.

An important work in this field is described in [16].

2) Yaw Commander: The "yaw commander" module, see Fig. 3, calculates de yaw reference depending on the current swarm agent's mission. This module could potentially decide in which direction to look in order to explore or to get the most localization information from the environment. These possibilities will be explored in future work.

3) Mission Planner: The mission planner allows the operator to define a mission as a set of separate tasks; which are, in turn, fully described by a set of numeric parameters. The mission definition requires a xml file where the mission is defined. It has different available tasks such as: take off, land, hover, sleep or move.

This module interacts with the trajectory planner module, the localization module when moving and with the hypothalamus module.

4) Basic Intelligence - Hypothalamus: This module implements low-level intelligence such as: monitoring the state of the other modules of the swarm agent and presenting a simple interface between the mission planner and other modules. Some of the commands that the mission planner can achieve through the hypothalamus module are: setting up the whole system to an active flying behavior (including the start-up of all other modules and the trajectory controller); and also simpler commands such as take-off, land or hover.

This module also implements the communication between its swarm agent and the rest of the swarm. Currently the communication is limited to sharing each drone's current pose.

E. Human-Robots Interface

A Human-Robot Interface was implemented in order to have a better understanding of the flight dynamics and the response of the drones in each mission. It can be used in real-time flight tests as well as simulated environments. The intended purposes of the interface are:

- To visualize and keep track of the progress of the drones in real-time during missions and send commands, if necessary.
- To verify that the modeled behavior and the expected performance of the different modules are adequate for executing the assigned mission in a simulated environment.
- To extend the assessment behavior and expected performance on types of missions that cannot be tested.
- To extrapolate to scenarios not currently available due to hardware or equipment limitations (higher number of drones).

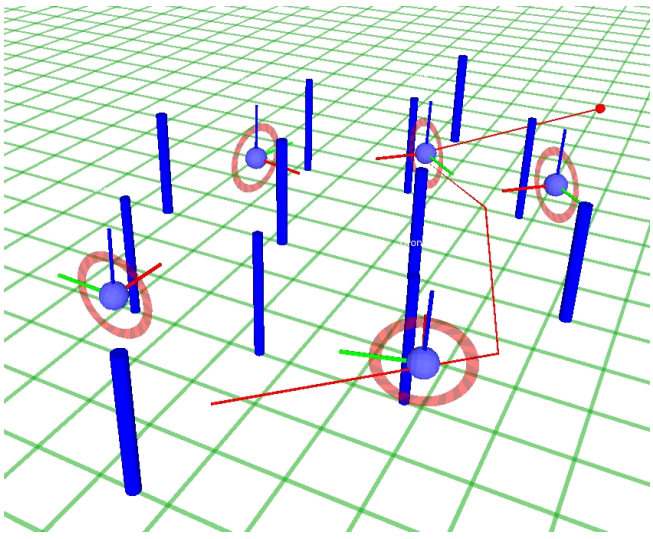

Fig. 4. Visualization of a simulated 5 drone flight in the Pinball Mission using the Rviz Interface. Drone axes: red x axis, green y axis and blue z axis. The blue cylinders represent the map columns and the green grid the floor. The red line shows the current planned trajectory for the selected drone and the red dot is the current mission point.

The interface developed, shown in figure 4, is based on the 3D visualization tool for ROS, Rviz. Communication is done in real time between all the ROS nodes. The 6D poses of the drones and the position of the obstacles are received via ROS topics as well as being able to send velocity, position and trajectory commands in real-time. The data collected during simulations has the purpose of validating the developed trajectory controller and mission planner in different scenarios.

IV. Simulation Results

Before a multi-aerial-robot flight all the hardware needs to be checked and set up, including not only the quadrotor platforms but also the test environment, the external visualization and processing computers, the WLAN, etc. Another very important requirement, for security reasons, is to have one emergency pilot per quadrotor. All this means that a lot of time and people are required to set up everything for an experimental test, which also means that every test has a significant cost. Since preparing a multi-aerial-robot system for a flight is costly, real tests should preferably be carried out after the system is tested in simulation.

Thanks to the modularity of our proposed system, it is possible to replace actual modules with simulated counterparts, since they only need to somewhat mimic the original module's interface. Then, except maybe for the modules that directly interface with the drone or the environment, the rest of the system can be tested in simulation. The value of such simulations is that they allow to test sets of modules of the actual flight architecture, easing the testing and debugging software development processes.

In order to simulate the basic instance of the framework, described in section III, two simulator modules were developed. The first one replaces the interface and approximates the flight dynamics of the quadrotor, which corresponds to the drone interface module described in subsection III-A. Since the Visual Marker Detector module implementation uses an external open-source library, instead of obtaining simulated images that could be processed by the Visual Marker Detector module, it was decided that the best option was to develop a simple simulated version of the whole module with a comparable marker detection and position estimation capabilities.

The following sections of the paper present the execution of a mission called: "The Pinball". It is noted that these two simulator modules were not designed to provide an accurate behavior compared to their real counterparts' behavior. The goal was to test the rest of the system in order to debug it and improve it, and also to check the mission specifications before experimental flights. The reader is invited to visit the research group's webpage to watch videos of these simulations and the corresponding experimental flights: `http://www.vision4uav.com/?q=quadrotor_stack`.

In order to interpret the simulation and experimental flight figures, the modules specifications have to be taken into account, which are described in the Arhictecture Software Modules section (Sec. II). From them, the overall expected behavior of the drones is:

- Each drone follows a sequence of waypoints given by its mission specification. The mission planner executes it sequentially.
- During execution, the planner will attempt to find trajectories that are collision-free. If it fails, the drone will be controlled to stay in the current position.
- If other swarm agents enter the current trajectory, the planner will detect it and will stop the drone and attempt to find a collision-free trajectory.
- If another swarm agent is on the current goal waypoint, the drone is commanded to stay in the current position. A new trajectory is planned when the goal position is again free.

A. Mission: The Pinball

This mission is carried out in a volleyball court (dimensions: 9 m x 18 m). Twelve poles (four of them with a diameter of 40 cm; and the other eight of 30 cm) are spread in the court with a distribution 1-3-4-3-1, shown in figures 5 and 6. These poles act like obstacles in the same way that the pins in the pinball. Five drones execute a navigation mission that consist on crossing the obstacles zone from one side of the court to the other. The mission specification consists on the following sequence of tasks:

1) Start the architecture operation and take-off,
2) Move to a goal landing location,
3) Land and stop the architecture.

In figures 5 and 6, the trajectories followed by each drone in two simulations of the same mission are displayed. As there is no high-level intelligence that synchronizes the swarm their behaviors are not deterministic and, thus, are different in every simulation execution:

- Simulation 1, Fig. 5: each drone followed a short and direct path, with very small rectifications, to its goal landing location.
- Simulation 2, Fig. 6: some drones followed a short and direct path, but others accomplished a very long path to avoid the other drones in the court.

Since the swarm agents plan their trajectories with no interaction with a global synchronization intelligence, the different executed paths are a demonstration of a low-intelligence swarm behavior.

V. Experimental Results

Once the system has been tested in simulation, the following step is to test it in a real experiment.

As the simulations were not an accurate representation of the reality, in the real tests, some issues that are not present in simulation appeared, such as:

- Inaccurate quadrotor model
- Inaccurate drone's sensors measurements
- Inaccurate performance of the computer vision algorithms

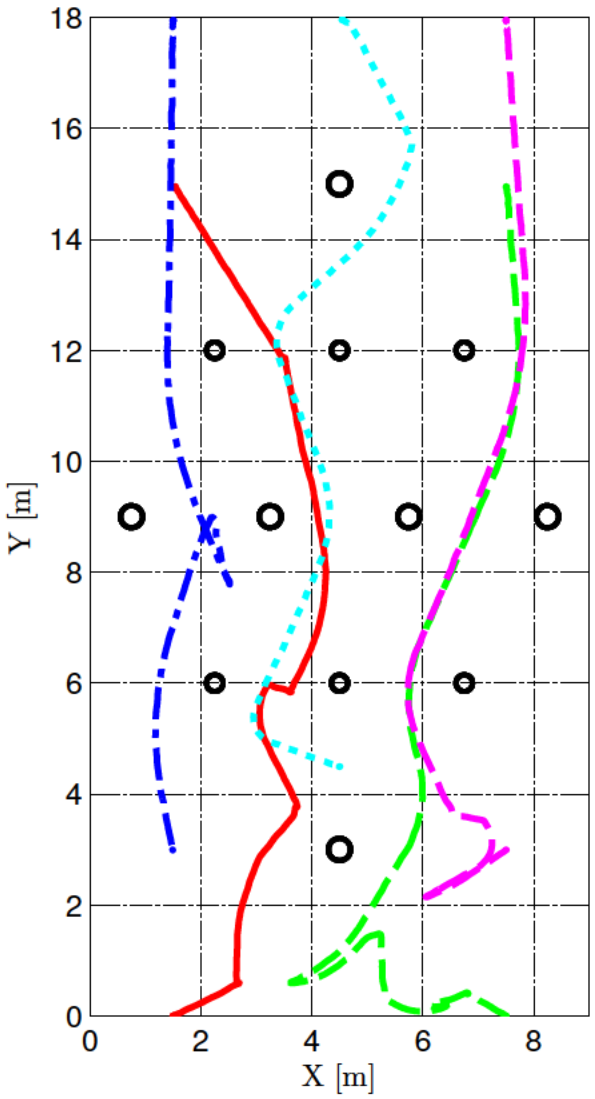

Fig. 5. Simulated flight of the Pinball Mission, where five drones flew autonomously. The executed trajectories of the drones are shown as line plots. The black circles are the poles. In this simulation, all the trajectories followed by the drones are direct and short. Some drones needed to create a longer path to avoid a collision. For example, the drone plotted in green had to go to the left of the map at the beginning of the simulation to avoid a collision with the drone in cyan. Once the collision was avoided, the drone replanned its trajectory again in order to avoid a collision with the red drone.

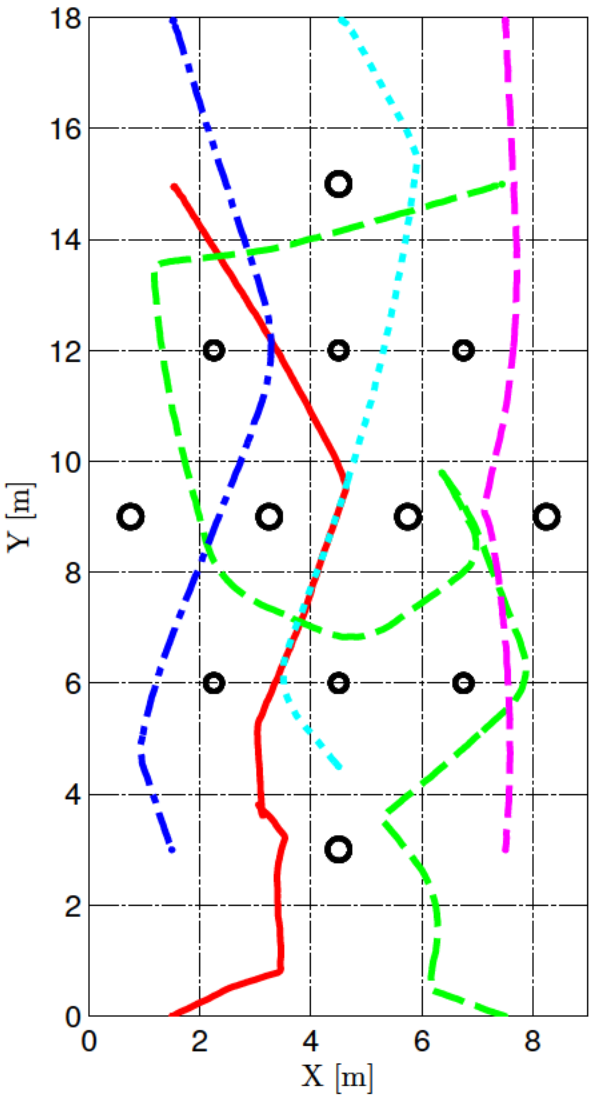

Fig. 6. Simulated flight of the Pinball Mission, where five drones flew autonomously. The figure can be interpreted similarly to Fig. 5. In this case, all the drones followed a short and direct path except the green one, which had to fly a very long path to avoid several collisions: first, with the cyan drone; and later with the magenta and red ones.

Those inaccuracies involve localization and control errors that the system has to minimize and deal with.

In this section, real tests of the mission simulated in section IV are achieved.

A. *Mission: The Pinball*

This is the same mission defined in section IV-A. Figure 7 shows the trajectories followed by each drone. These trajectories are more noisy that the one achieved through the simulations due to the problems and inaccuracies cited above. However, the system keeps working correctly and completes the mission.

VI. Future Work

There are many possible ways to research in multi-aerial-robot based on the presented architecture. Thanks to the modularity of the system, specialists in different robotics fields can contribute to specific modules of the architecture. Since the system is open-source and it is fully working both in simulation and in reality, each specialist can download it and develop a module for the system that attracts their interest and, afterwards, he will be able to test it against the rest of the system in simulation or in experimental flights. Some possible work lines are described in the following paragraphs:

One possibility is to use the described modules to research in the fields of swarming and multi-robot systems. More complex behaviours could be added to the agents of the swarm in order to accomplish more complex missions that

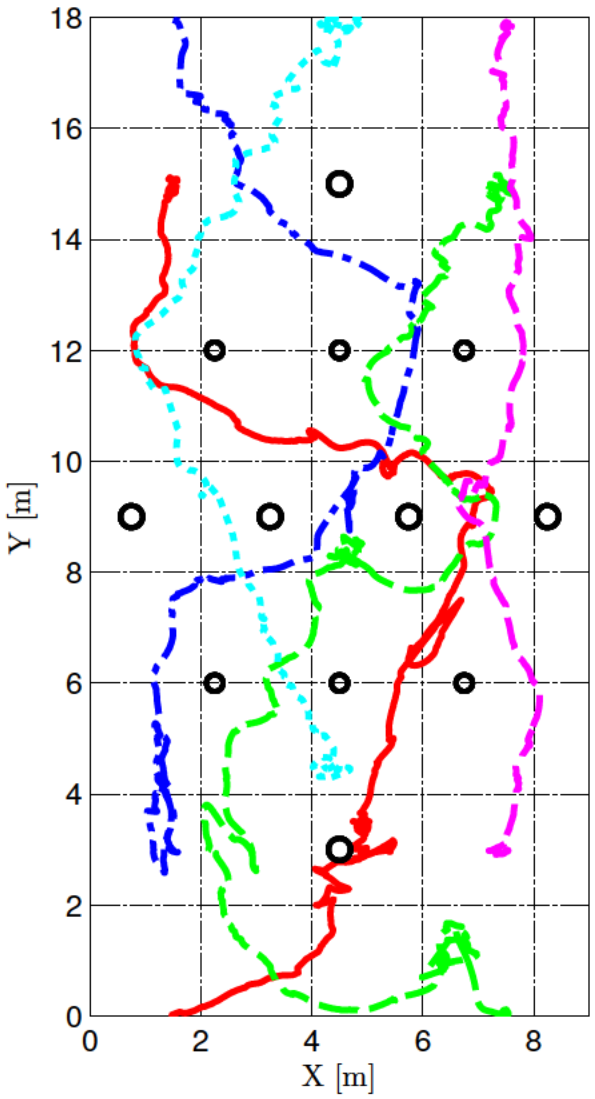

Fig. 7. Experimental flight of the Pinball Mission, where five drones flew autonomously. The executed trajectories of the drones are shown as line plots. The black circles are the poles. Some of the drones followed a direct and short path (blue, magenta, and cyan) and others a longer one (red and green). The experimental trajectories are noisier than the simulated ones, due to limitations in the simulator.

require a higher level of coordination. Or, putting behind the swarming aspect, a central unit that coordinates the robots can be added in order to improve their synchronization.

Another possibility is to research in the field of fault tolerant systems, creating a module that is able to detect events such as component failures (i.e. one of our modules), and after its occurrence attempt to rearrange the module architecture excluding the malfunctioning module.

Other future work that we have in mind is to work with heterogeneous swarm of drones. That means that the system could be composed of some cheap drones, which are able to achieve simple tasks, and other more expensive drones, which are in change of more demanding tasks. This alternative is feasible thanks to the modularity and the multiplatform orientation of the system.

Although the system works correctly, it can only perform navigation tasks and it still requires to fly in an structured environment. This means that the mission planner could be redesigned in order to obtain a more versatile swarm behaviour. Better perception, localization and mapping algorithms and sense and avoid algorithms that do not rely on visual markers are needed to move the experiments out of the laboratory.

VII. Conclusions

This paper presented a fully functional framework for the research of multirotor swarms and of multi-aerial-robot systems. Thanks to the modularity in the design of the system, its modules could be easily replaced by new ones with other features. The system is also multiplatform, allowing the use of diverse robotic platforms that use different modules depending on their characteristics. The framework has been developed in C++, under the standard C++11, and using the worldwide utilized ROS to execute and communicate the different modules of the system.

This paper also presented a fully working instance of the framework that allows the autonomous navigation of a swarm of the cost-effective platform ARDrone 2.0 in structured environments. Our system has a mid-level controller that lets the drone to follow the free-collision trajectories given by the trajectory planner included in the high-level control. This high-level control also consists of a basic mission planner and a basic intelligence that we call hypothalamus. For the localization and mapping task, the system relies on the drone's odometry and on a several visual markers named ArUcos. The system counts also with a simulator to test the functionality of the system in a less time consuming way.

In order to contribute with the scientific development, we made the framework and our module's implementation open-source, giving anyone interested in the field of the aerial robotics system a fully-operative starting point to develop their own algorithms. The code can be downloaded from the research group webpage: `http://www.vision4uav.com/?q=quadrotor_stack`.

We are aware that our module's implementation is quite basic, but a huge number possible future works are open and easily achievable thanks to the design of the framework. Our main interests are the developing of heterogeneous swarms and the improvement of our modules such as the localization and mapping that allow the navigation of unstructured environments without visual markers.

The presented implementation of the system has not only been tested under simulation or in real tests, but also, it was awarded the First Prize in the category Indoors Autonomy of the International Micro Air Vehicle Indoor Flight Competition (IMAV 2013) [3].

References

[1] ardrone autonomy ros stack. `https://github.com/AutonomyLab/ardrone_autonomy/`.

[2] Aruco: a minimal library for augmented reality applications based on opencv. `http://www.uco.es/investiga/grupos/ava/node/26`.

[3] Imav 2013 flight competition rules. http://www.imav2013.org/index.php/information.

[4] Parrot ardrone 2.0 web. http://ardrone2.parrot.com/.

[5] Ros web. http://www.ros.org/wiki/.

[6] *The Navigation and Control Technology Inside the AR.Drone Micro UAV*, Milano, Italy, 2011.

[7] G. Conte. *Vision-Based Localization and Guidance for Unmanned Aerial Vehicles*. PhD thesis, Linkopings universitet, 2009.

[8] R. Motwani D. Hsu, J.C. Latombe. Path planning in expansive configuration spaces. In *Proceedings of the IEEE International Conference on Robotics and Automation, pp. 27192726*, 1997.

[9] P. de la Puente, D. Rodriguez-Losada, L. Pedraza, and F. Matia. Robot goes back home despite all the people. In *Proc. 5th. Conference on Informatics in Control, Automation and Robotics ICINCO 2008 Funchal, Portugal*, pages 208–213, 2008.

[10] P. de la Puente, D. Rodriguez-Losada, and A. Valero. 3D Mapping: testing algorithms and discovering new ideas with USARSim. In *USARSim workshop, IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[11] P. de la Puente, D. Rodriguez-Losada, A. Valero, and F. Mata. 3D feature based mapping towards mobile robots enhanced performance in rescue missions. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[12] Ayssam Elkady and Tarek Sobh. Robotics middleware: A comprehensive literature survey and attribute-based bibliography. *Journal of Robotics*, 2012, 2012.

[13] J. Engel, J. Sturm, and D. Cremers. Camera-based navigation of a low-cost quadrocopter. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[14] J. Faigl, T. Krajník, J. Chudoba, M. Saska, and L. Přeučil. Low-Cost Embedded System for Relative Localization in Robotic Swarms. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2013.

[15] Willow Garage. Ros: Robot operating system. www.ros.org/.

[16] Daniel Hennes, Daniel Claes, Wim Meeussen, and Karl Tuyls. Multi-robot collision avoidance with localization uncertainty. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 147–154. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[17] L. Jayatilleke and N. Zhang. Landmark-based localization for unmanned aerial vehicles. In *IEEE International Systems Conference (SysCon'13)*, pages 448–451, 2013.

[18] A. Kapoulkine. pugixml. http://pugixml.org/.

[19] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic, 1991.

[20] G. Mao, S. Drake, and B. D. O. Anderson. Design of an Extended Kalman Filter for UAV Localization. In *Information, Decision and Control, 2007 (IDC'07)*, pages 224–229, 2007.

[21] B. Raphael P. E. Hart, N. J. Nilsson. A formal basus for the heuristic determination of minimum cost paths. *IEEE Transactions on SYstems Science and Cybernetics*, 4(2):100–107, 1968.

[22] Jesús Pestana. On-board control algorithms for Quadrotors and indoors navigation. Master's thesis, Universidad Politécnica de Madrid, Spain, 2012.

[23] Jesús Pestana, Ignacio Mellado-Bataller, Jose Luis Sanchez-Lopez, Changhong Fu, Iván F Mondragón, and Pascual Campoy. A general purpose configurable controller for indoors and outdoors gps-denied navigation for multirotor unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, 73(1-4):387–400, 2014.

[24] P. Rudol. Increasing autonomy of unmanned aircraft systems through the use of imaging sensors. Master's thesis, Linkoping Institute of Technology, 2011.

[25] Jose Luis Sanchez-Lopez, Jesús Pestana, Paloma de la Puente, Adrian Carrio, and Pascual Campoy. Visual quadrotor swarm for the imav 2013 indoor competition. In Manuel A. Armada, Alberto Sanfeliu, and Manuel Ferre, editors, *ROBOT2013: First Iberian Robotics Conference*, volume 253 of *Advances in Intelligent Systems and Computing*, pages 55–63. Springer, 2013.

[26] J. De Schutter, J. De Geeter, T. Lefebvre, and H. Bruyninckx. *Kalman Filters: A Tutorial*, 1999.